

Parameterized Algorithms

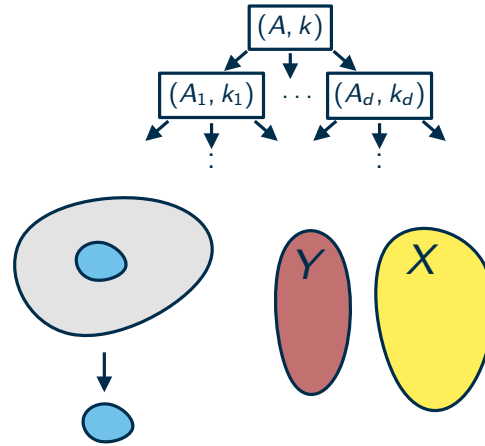
Lower Bounds: Reductions and the W-Hierarchy

Thomas Bläsius

Content

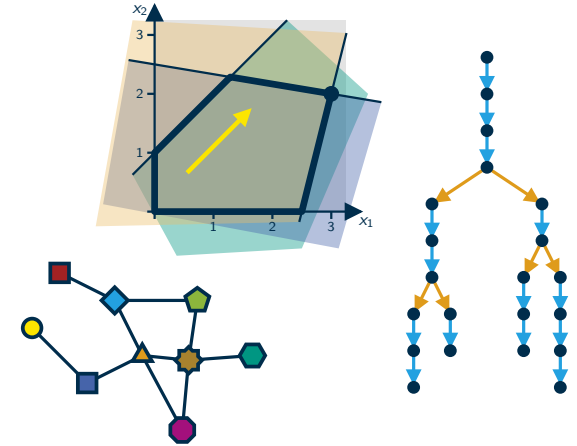
Basic toolbox

- bounded search trees
- kernelization
- iterative compression



Extended toolbox

- linear programs
- branch-and-reduce
- color coding



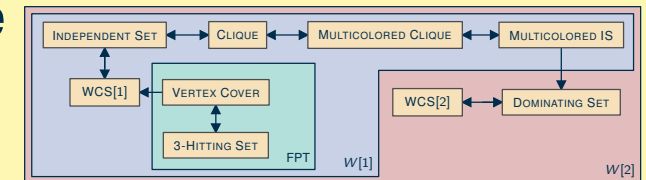
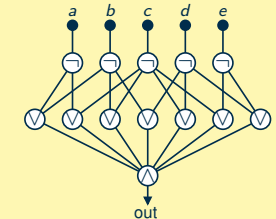
Tree width

- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

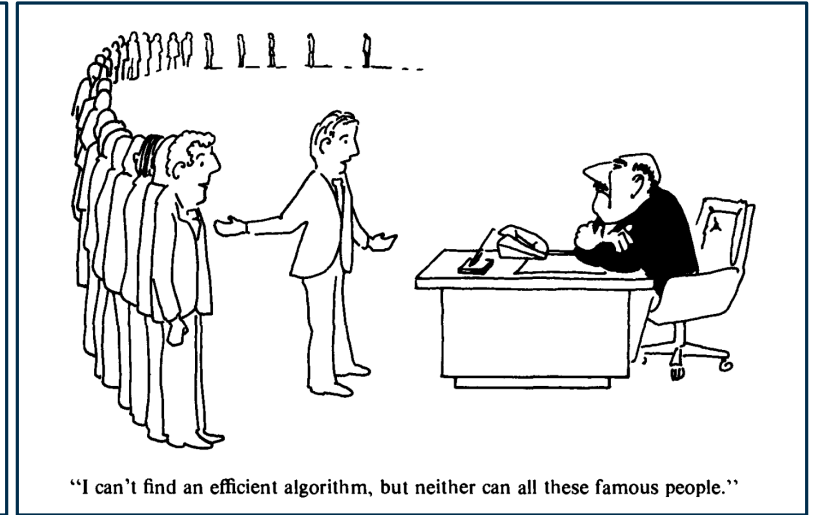
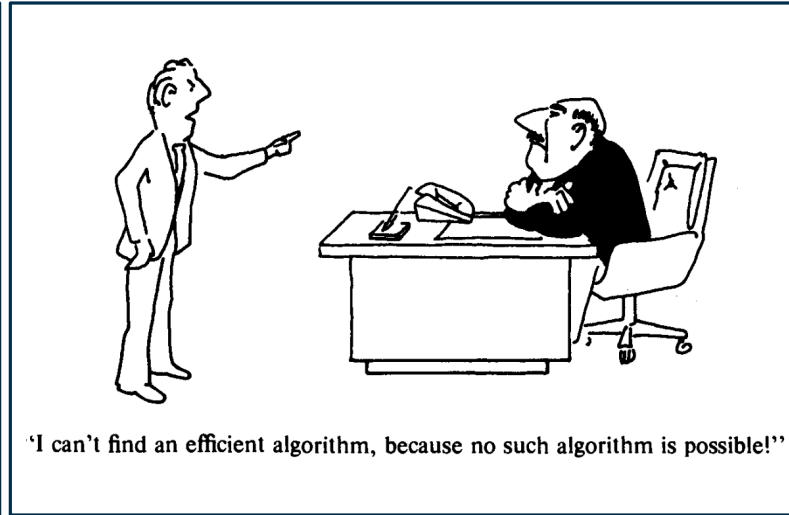
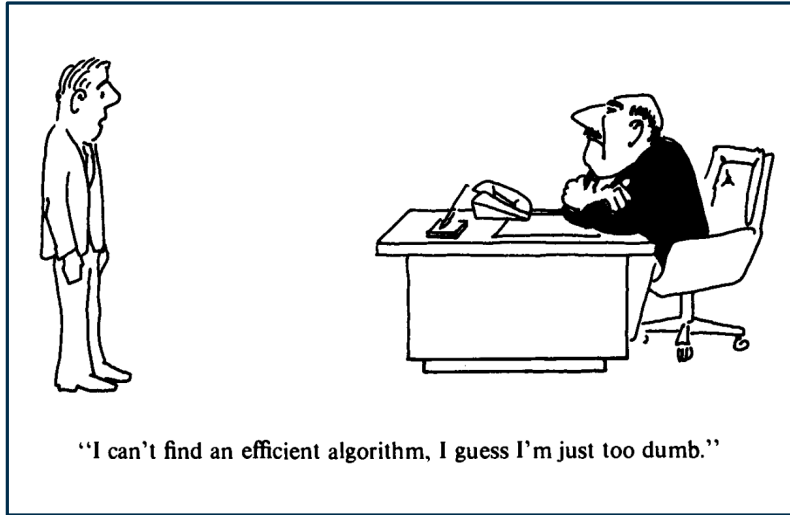


Lower bounds

- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH



Why do we want lower bounds?



Problem

- non-existence of an algorithm is difficult to prove

Solution

- show: my problem efficiently solvable $\Rightarrow$ a known hard problem efficiently solvable
- tool: reduction between problems

Polynomial reductions

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

- map each instance I of $\mathcal{L}$ to an instance I' of $\mathcal{L}'$
- I is yes-instance $\Leftrightarrow I'$ is yes-instance
- the map must be computable in polynomial time

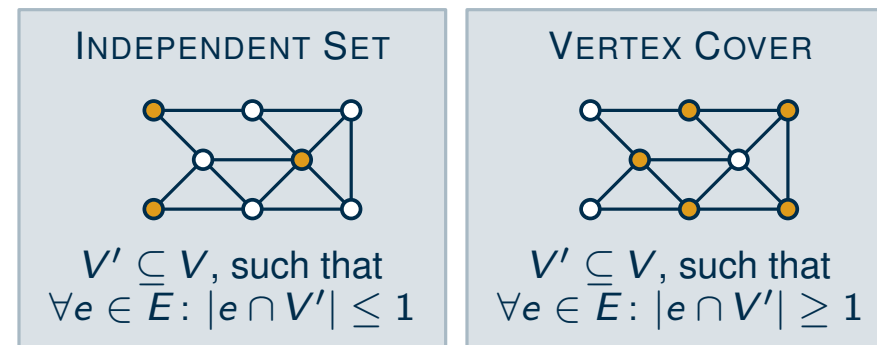
Implication

- polynomial algorithm for $\mathcal{L}' \Rightarrow$ polynomial algorithm for $\mathcal{L}$
- my problem $\mathcal{L}'$ efficiently solvable $\Rightarrow$ known hard problem $\mathcal{L}$ efficiently solvable

(for “efficient” = “polynomial”)

Example: INDEPENDENT SET to VERTEX COVER

- reduction: G has IS of size $k \Leftrightarrow G$ has VC of size $n - k$
- thus: $VC \in P \Rightarrow IS \in P$
- And “ $VC \in FPT \Rightarrow IS \in FPT$ ”? $\rightarrow$ parameter too large
- for “efficient” = “FPT”, we need a different reduction



Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow (I', k')$ yes-instance
- the map must be computable in FPT-time $f(k) \cdot |I|^{O(1)}$

(f and g computable)

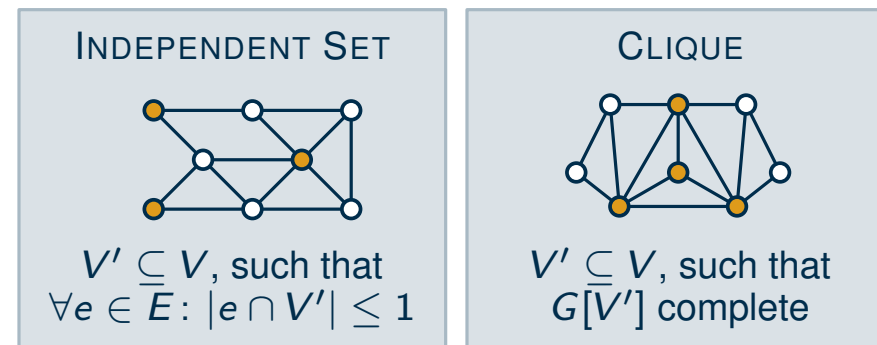
Implication

- FPT algorithm for $\mathcal{L}' \Rightarrow$ FPT algorithm for $\mathcal{L}$
- my problem $\mathcal{L}'$ efficiently solvable $\Rightarrow$ known hard problem $\mathcal{L}$ efficiently solvable

(for “efficient” = “FPT”)

Example: INDEPENDENT SET to CLIQUE

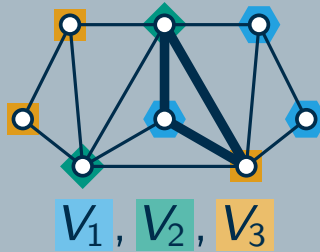
- G has IS of size $k \Leftrightarrow$ complement of G has k -clique
- thus: CLIQUE $\in$ FPT $\Rightarrow$ IS $\in$ FPT
- inverse also holds: CLIQUE $\in$ FPT $\Leftrightarrow$ IS $\in$ FPT
- conjecture: CLIQUE, IS $\notin$ FPT



CLIQUE $\rightarrow$ MULTICOLORED CLIQUE

Problem: MULTICOLORED CLIQUE

Given a graph $G = (V, E)$, a parameter k and a partition $(V_1, \dots, V_k)$ of the vertices. Is there a clique with exactly one vertex from each V_i ?



Reduction: CLIQUE to MULTICOLORED CLIQUE

- replace $v \in V$ by $v^1, \dots, v^k$ with $v^i \in V_i$
- for $uv \in E$ connect u^i with v^j for all $i \neq j$
- set $k' = k$

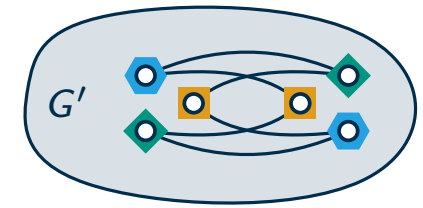
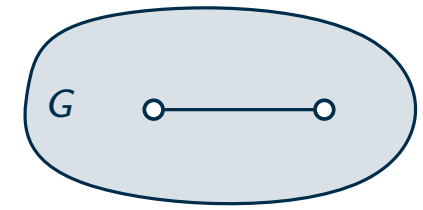
k -clique in $G \Rightarrow$ colorful k -clique in G'

- let $v_1, \dots, v_k$ form a clique in G
- then $v_1^1, \dots, v_k^k$ is a clique in G'
- all vertices have different colors

colorful k -clique in $G' \Rightarrow k$ -clique in G

- colorful clique: $v_{\pi(1)}^1, \dots, v_{\pi(k)}^k$ with $\pi: [k] \rightarrow [n]$
- $v_{\pi(1)}, \dots, v_{\pi(k)}$ form a clique in G
- it has size k as π is injective

instance $(G, 3)$ of CLIQUE

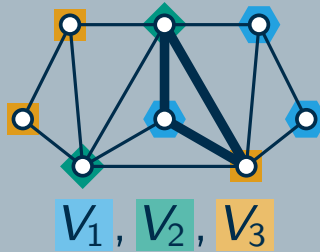


instance of MC CLIQUE: $(G', 3, (V_1, V_2, V_3))$

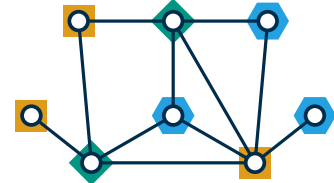
MULTICOLORED CLIQUE $\rightarrow$ CLIQUE

Problem: MULTICOLORED CLIQUE

Given a graph $G = (V, E)$, a parameter k and a partition $(V_1, \dots, V_k)$ of the vertices. Is there a clique with exactly one vertex from each V_i ?



instance of MC CLIQUE: $(G, 3, (V_1, V_2, V_3))$



instance $(G', 3)$ of CLIQUE

Reduction: MULTICOLORED CLIQUE to CLIQUE

- delete edges between equally colored vertices
- set $k' = k$

colorful k -clique in $G \Rightarrow k$ -clique in G'

- no edges of the colorful clique are deleted
- it remains a clique of size k in G'

k -clique in $G' \Rightarrow$ colorful k -clique in G

- no vertices with equal color are adjacent in G'
- each clique in G' is colorful in G

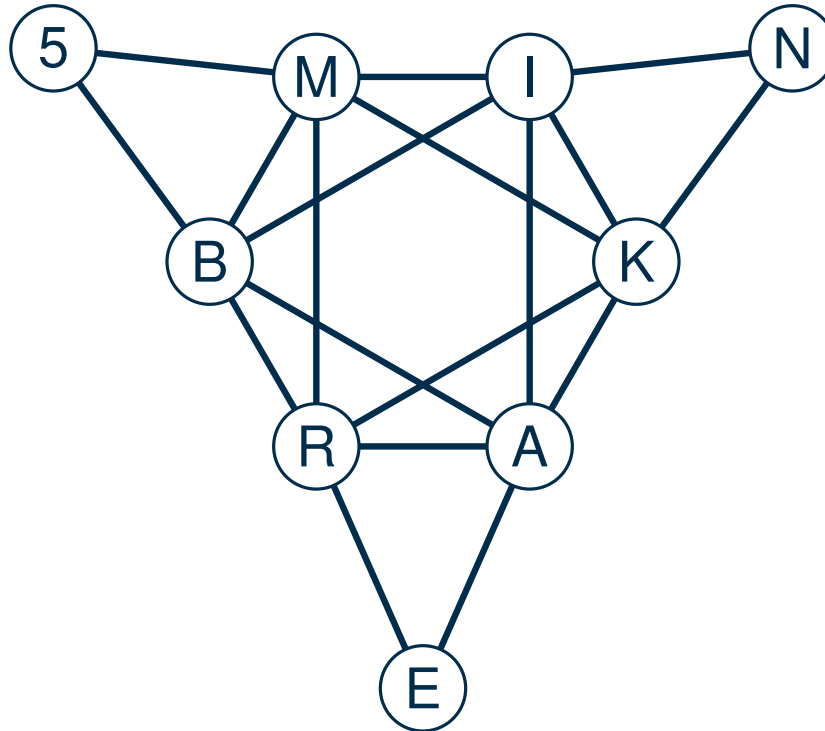
Reductions so far



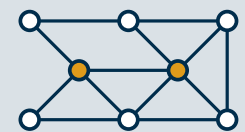
DOMINATING SET

Goal

- select as few vertices as possible
- for every vertex: the vertex or one of its neighbors is selected



DOMINATING SET



$V' \subseteq V$, such that
 $\forall v \in V: |N[v] \cap V'| \geq 1$

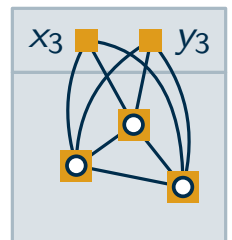
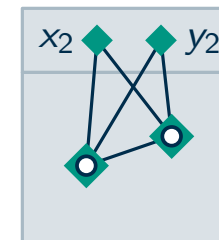
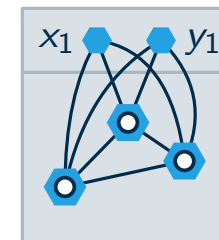
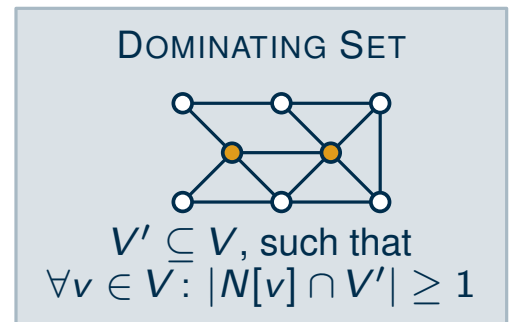
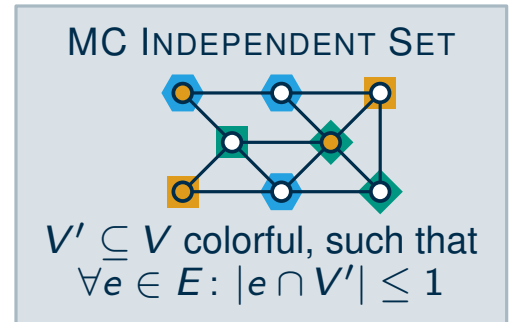
DOMINATING SET

Reduce MULTICOLORED INDEPENDENT SET to DOMINATING SET

- enforce the following properties in an instance of DOMINATING SET
 - exactly one vertex of each color must be chosen ✓
 - forbid the choice of certain vertex pairs

Exactly one vertex for each color

- vertex x_i with edges to all vertices in V_i → must choose one of $V_i \cup \{x_i\}$
- problem: one could choose x_i
- solution: additional vertex y_i → choosing x_i and y_i is too expensive
- make it sufficient to choose one of V_i : clique on V_i



DOMINATING SET

Reduce MULTICOLORED INDEPENDENT SET to DOMINATING SET

- enforce the following properties in an instance of DOMINATING SET
 - exactly one vertex of each color must be chosen
 - forbid the choice of certain vertex pairs

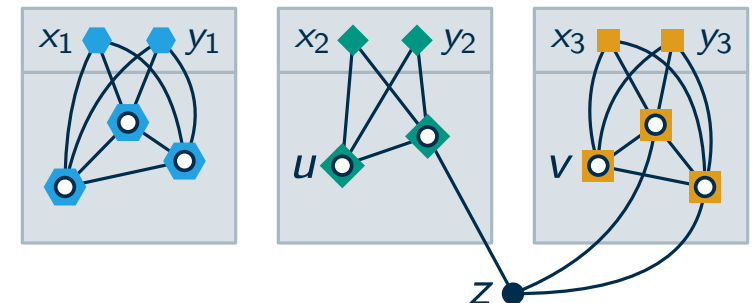
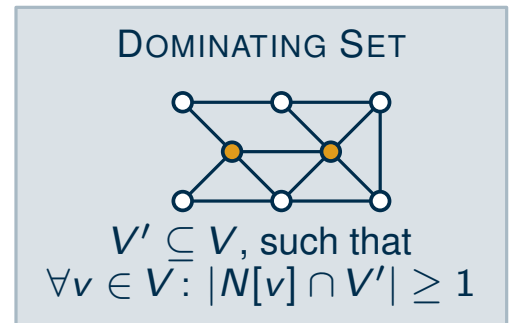
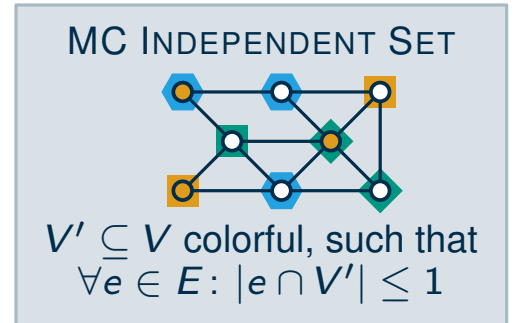


Exactly one vertex for each color

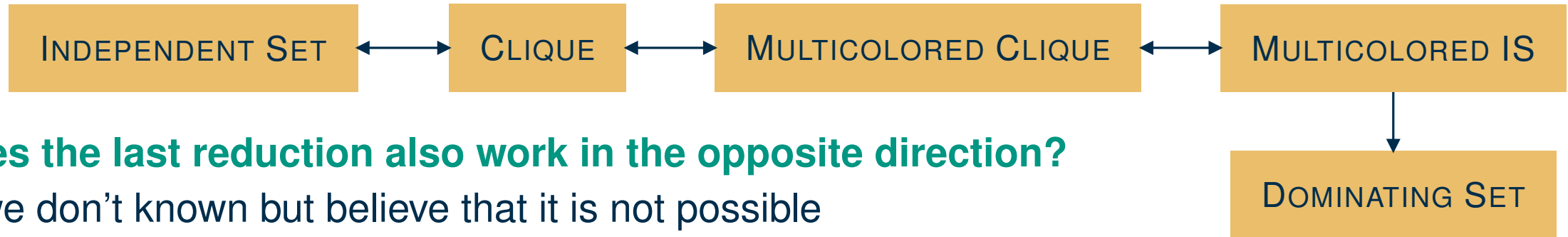
- vertex x_i with edges to all vertices in V_i → must choose one of $V_i \cup \{x_i\}$
- problem: one could choose x_i
- solution: additional vertex y_i → choosing x_i and y_i is too expensive
- make it sufficient to choose one of V_i : clique on V_i

Forbid to choose u and v at the same time

- idea: add vertex z that is covered unless u and v are chosen
- connect z with all vertices of u 's and v 's color, except u and v



Reductions so far



Does the last reduction also work in the opposite direction?

- we don't know but believe that it is not possible
- DS seems strictly harder than CLIQUE or IS ($DS \in FPT \Rightarrow IS \in FPT$, but not the other way round)
- we need a more refined notion of hardness

Comparison with complexity class P

- here usually one notion of hardness suffices: NP-hardness
- there are intermediate problems (assuming $P \neq NP$), but we don't know any natural cases
(candidates: prime factorization, graph isomorphism)

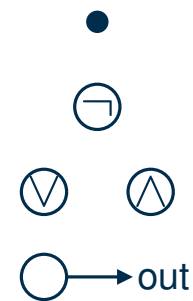
What now?

- define hierarchy of complexity classes
- use prototypical complete problems for each level of the hierarchy

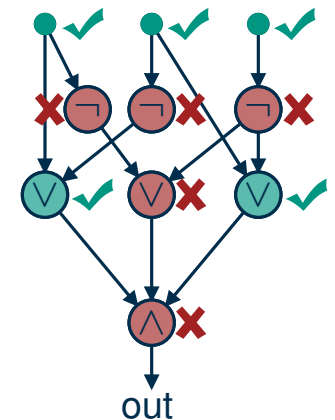
WEIGHTED CIRCUIT SATISFIABILITY

Boolean circuit

- directed acyclic graph (DAG) with the following types of nodes:
 - sources (in-degree 0) are **input nodes**
 - **negation nodes** have in-degree 1
 - **and nodes** and **or nodes** have in-degree ≥ 2
 - one sink (out-degree 0) is the **output node**
- a assignment of the input nodes to 0 or 1 yields values for all other nodes
- an assignment is **satisfying** if the output node has value 1
- **weight** of an assignment = number of 1s for input nodes



(in the obvious way)



Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)

Given a Boolean circuit and a parameter k , is there a satisfying assignment of weight exactly k ?

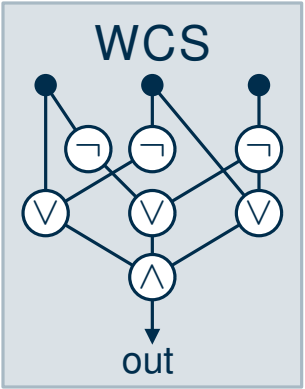
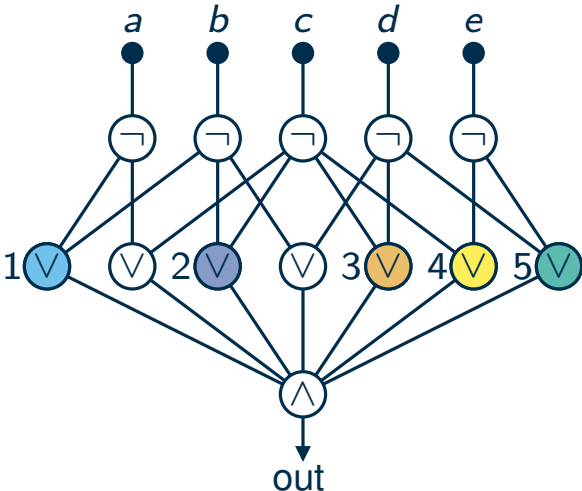
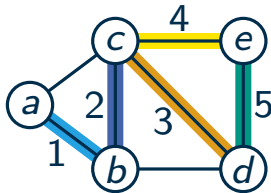
Reductions as far as the eye can see

INDEPENDENT SET → WCS

INDEPENDENT SET

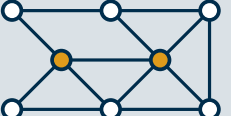


$V' \subseteq V$, such that
 $\forall e \in E: |e \cap V'| \leq 1$

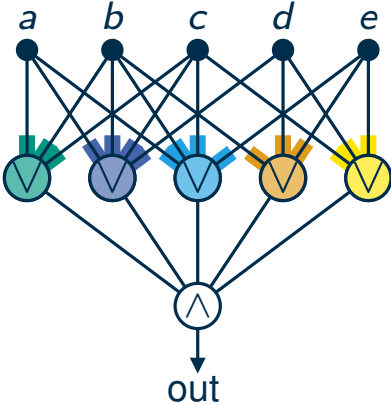
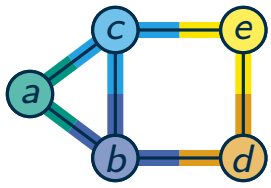


DOMINATING SET → WCS

DOMINATING SET

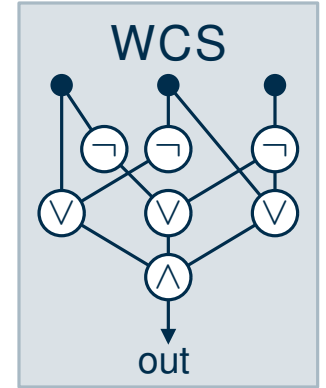
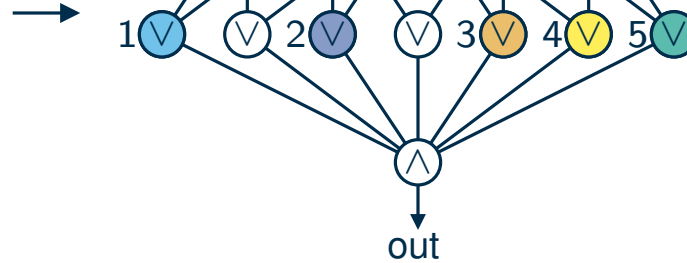
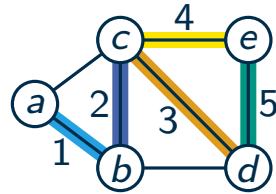
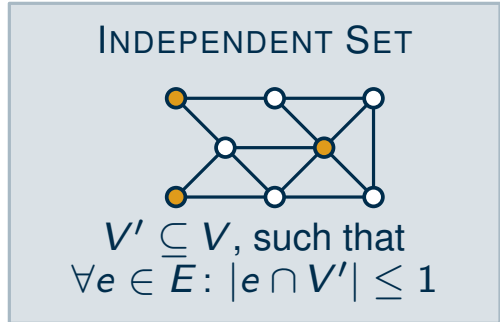


$V' \subseteq V$, such that
 $\forall v \in V: |N[v] \cap V'| \geq 1$

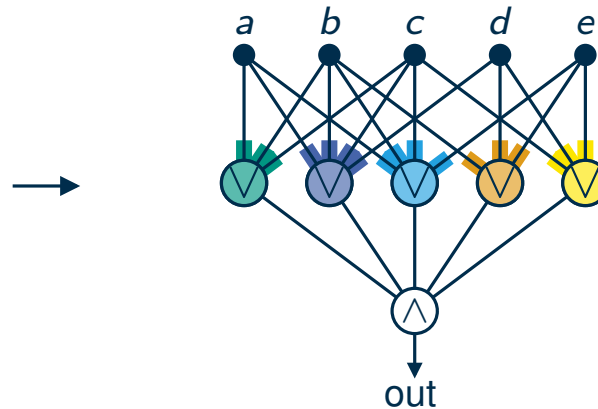
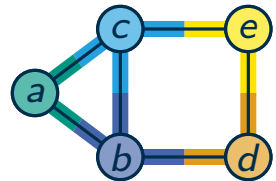
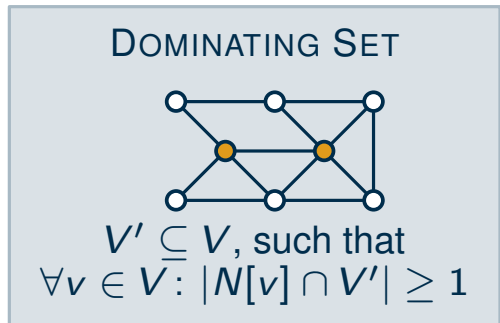


Reductions as far as the eye can see

INDEPENDENT SET $\rightarrow$ WCS



DOMINATING SET $\rightarrow$ WCS



Observations

- both circuits: constant height
- the DS circuit has more nodes with in-degree > 2
- Does this make DS harder (with respect to FPT) than IS?

Weft

Definition

The largest number of nodes with in-degree > 2 on a directed path is called **weft** of the circuit.

Problem

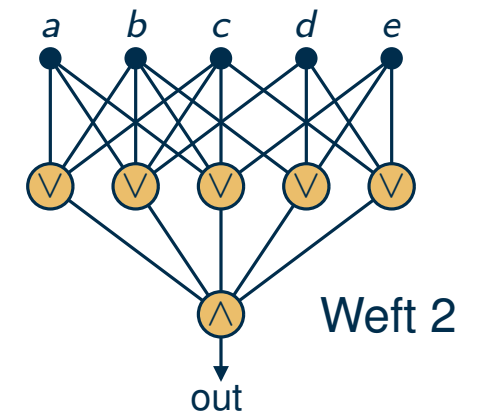
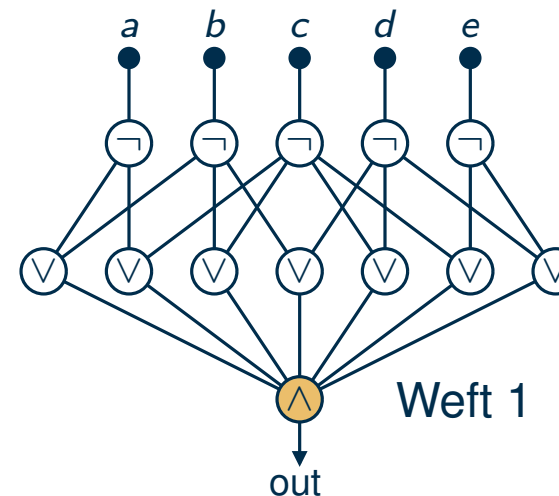
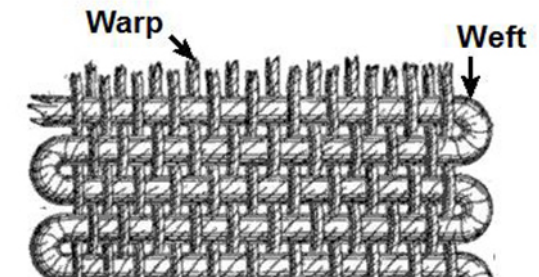
WCS $[t]$ restricts WCS to circuits of constant height and weft $\leq t$.

Definition

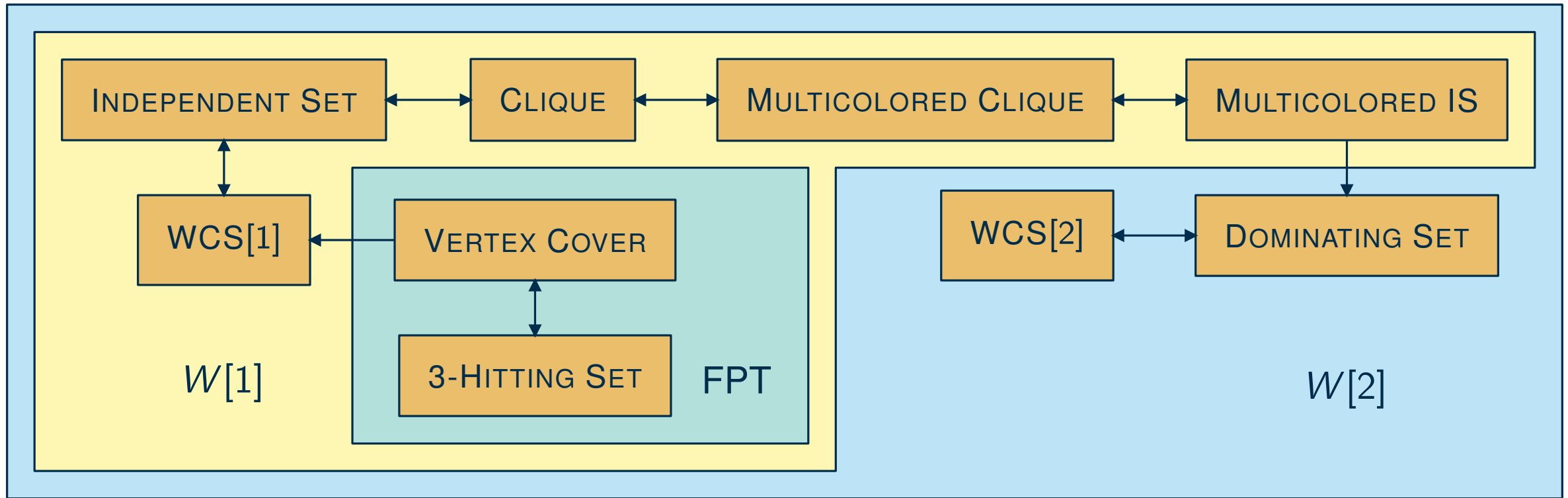
The class **W** $[t]$ is the set of problems that have a parameterized reduction to **WCS** $[t]$.

Just seen

- INDEPENDENT SET $\in W[1]$
- DOMINATING SET $\in W[2]$



Bisherige FPT-Reduktionen



Additional reductions

- $WCS[1] \rightarrow IS$ and $WCS[2] \rightarrow DS$
- every $W[1]$ -problem reduces to IS $\rightarrow W[1]$ -complete
- analogously: DS is $W[2]$ -complete

Relations between classes

- $W[1] \subseteq W[2]$
- $FPT \subseteq W[1] \subseteq W[2]$

Why?

Why?

Wrap-Up

The W -hierarchy

- complexity classes $\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ defined via prototypical complete problem $\text{WCS}[t]$: $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ reducible to $\text{WCS}[t]$
(via parameterized reductions)
- we believe: each inclusion is strict

Is my $W[t]$ -completeness proof useless, if $W[t] = \text{FPT}$?

- still true: you don't find an FPT-algo due to a fundamental problem, not because you are stupid
- finding an FPT-algo for one complete problem $\rightarrow$ reductions yield additional FPT-algos

How do I prove that my problem is hard?

- reduce from other hard problems
- reductions from INDEPENDENT SET or CLIQUE prove $W[1]$ -hardness
- reductions from DOMINATING SET or HITTING SET prove $W[2]$ -hardness